Answer the following four questions:

Question No. 1 **(25 Marks)**

a) Consider the regular expression below which can be used as part of a specification of the definition of exponents in floating-point numbers. Assume that the alphabet consists of numeric digits ('0' through '9') and alphanumeric characters ('a' through 'z' and 'A' through 'Z') with the addition of a selected small set of punctuation and special characters (say in this example only the characters '+' and '-' are relevant). Also, in this representation of regular expressions the character '.' denotes concatenation.

Exponent = (+ | - | ε) . (E | e) . (digit)+

For this regular expression answer the following questions:

i. Derive an NFA capable of recognizing this language.

ii. Derive the DFA for the NFA that you derive in a.

b) Given the following context free grammar

$S \rightarrow Ab \mid aaB$

$A \rightarrow a \mid Aa$

$B \rightarrow b$

i. Explain why the given grammar is ambiguous.

ii. Find an equivalent unambiguous context-free grammar.

c) For the following grammar:

S ⟶ S and S | true

i. List 4 derivations for the string "true and true and true".

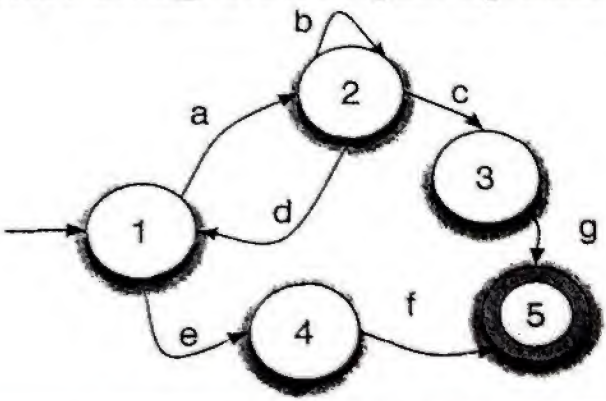
ii. Label each derivation as left-most, right-most, or neither.

iii. List the parse tree for each derivation

iv. What is implied about the associativity of "and" for each parse tree?

Question No. 2 **(25 Marks)**

a) Convert the following NFA to a regular expression :



b) Consider the following grammar:

S → bAb | bBa
A → aS | CB
B → b | BC
C → c | cC

i. Give two reasons why this grammar is not LL(1) →
ii. Rewrite the grammar, introducing as few new non-terminals as possible, so that it is LL(1) and recognizes the same strings as the original grammar.
iii. Construct the first and follow sets for each non-terminal in the rewritten grammar.
iv. Construct an LL(1) parse table for the rewritten grammar.

c) Below, on the left, is a description of several parts of a classic optimizing compiler. The names of those parts are found in the list on the right labelled with letters. Place the letter for the correct name next to each description.

1. optimizer a part of a compiler that is responsible for recognizing syntax.

2. parser a part of a compiler that takes as input a stream of characters and produces as output a stream of words along with their associated syntactic categories.

3. FrontEnd a phase of a compiler that focuses on understanding the source-language and encodes this knowledge by translating the program into an intermediate representation.

4. Sem ---------- a part of a compiler that understands the meanings of variable names and other symbols and checks that they are used in ways consistent with their definitions.

5. linker ---------- an intermediate representation -to-intermediate representation transformer that tries to improve the intermediate representation program in some way.

6. scanner ---------- a phase of a compiler that maps the intermediate representation program into the instruction set and the finite resources of the target machine.

A. back-end

B. semantic analysis

C. front-end

D. scanner

E. linker

F. optimizer

G. parser

Question No. 3 (20 Marks)

1. Consider a context-free language over the alphabet $\Sigma = \{1, 2, a, b\}$ recognized by the following recursive descent parser:

```
bool term(TOKEN tok) { return *next++ == tok; }
bool S() { TOKEN *save = next; return (next=save, N() && T() && S()) ||
           (next = save, N() && T())}

//Token for 1 is ONE, 2 is TWO
bool N() { TOKEN *save = next;return ((next = save), term(ONE)) ||
                                    ((next = save), term(TWO)); }

bool T() { TOKEN *save = next; return (next=save, A()) || (next=save, B()); }

bool A() { TOKEN *save = next; return (next=save, term(ATOK)) ||
           (next=save, T() && term(ATOK)); } // Token for a is ATOK

bool B() { TOKEN *save = next; return (next=save, term(BTOK)) ||
           (next=save, T() && term(BTOK)); } // Token for b is BTOK
```

i. Give the context-free grammar from which this parser was produced.
ii. Write a regular expression that recognizes the same language as the CFG.

2. Given the following grammar G for strings over the alphabet {a,b,+,*,(,)} with non-terminals E, T, and F where E is the start symbol:

$E \longrightarrow T \mid E + T$
$T \longrightarrow F \mid T * F$
$F \longrightarrow a \mid b \mid (E)$

i. Is the grammar G SLR?
ii. If it is SLR, Give the right most derivation and the stack configurations that show how the string: **a*b+b*(a+a)** is parsed.

Question No. 4 (30 Marks)

For each of the following, please circle the letter introducing the best answer. (Check all that apply.)

1. The regular languages that are equivalent to the regular language: $(0 + 1)^*1(0 + 1)^*$ are:

a) $(01 + 11)^*(0 + 1)^*$
b) $(0 + 1)^*(10 + 11 + 1)(0 + 1)^*$
c) $(1 + 0)^*1(1 + 0)^*$
d) $(0+ 1)^*(0 + 1)(0 + 1)^*$

2. Which of the following grammars are ambiguous?

a) $S \longrightarrow SS \mid a \mid b$
b) $E \longrightarrow E + E \mid id$
c) $E \longrightarrow E' \mid E' + E$
d) $E' \longrightarrow -E' \mid id \mid (E)$

3. The regular languages that are correct specifications of the English-language description given below:

Twelve-hour times of the form "04:13PM". Minutes should always be a two digit number, but hours may be a single digit.

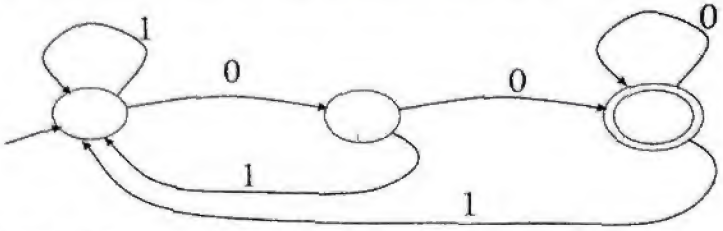
a) (0 + 1)?[0-9]:[0-5][0-9](AM + PM)

b) ((0 + ε)[0-9] + 1[0-2]):[0-5][0-9](AM + PM)

c) (0*[0-9] + 1[0-2]):[0-5][0-9](AM + PM)

d) (0?[0-9] + 1(0 + 1 + 2):[0-5][0-9](A + P)M

4. The regular language that denotes the same language as this finite automaton is:



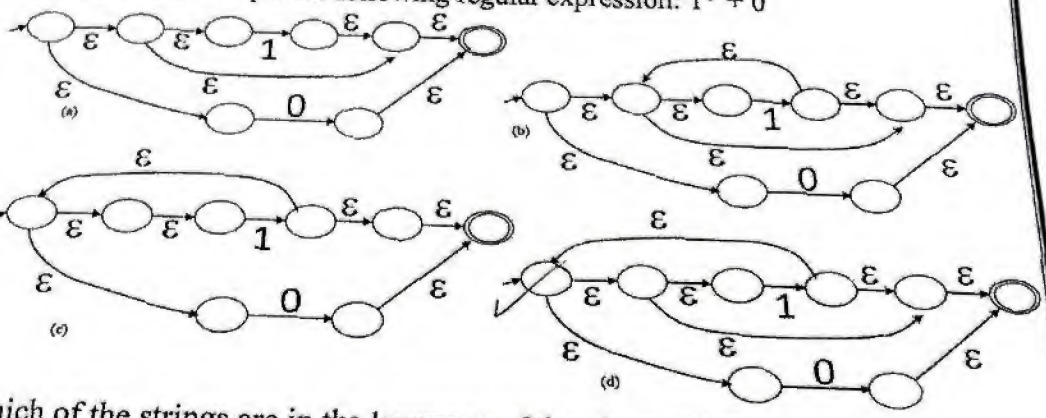
a) (0 + 1)*

b) (1* + 0)(1 + 0)

c) 1* + (01)* + (001)* + (000*1)*

d) (0 + 1)*00

5. The NFA that accepts the following regular expression: 1* + 0



5. Which of the strings are in the language of the given CFG?

$S \longrightarrow aXa$

$X \longrightarrow bY \mid \varepsilon$

$Y \longrightarrow cXc \mid \varepsilon$

a) abcba

b) acca

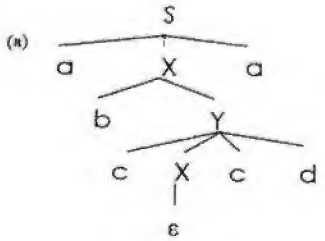
c) aba

d) abcbcba

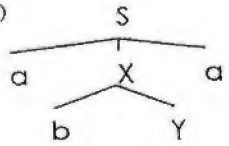
7. Which of the following is a valid parse tree for the given grammar?

S ⟶ aXa
X ⟶ bY | ε
Y ⟶ cXc | d | ε

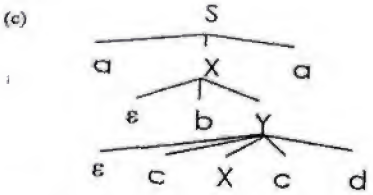
(a)



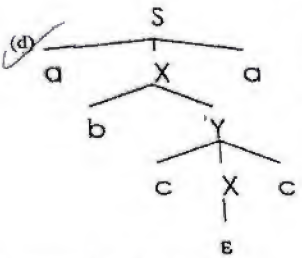
(b)



(c)



(d)



8. For the given grammar, what is the correct shift-reduce parse for the string: id + -id

E ⟶ E' | E' + E
E' ⟶ -E' | id | (E)

(a)

|id + -id
id|+ -id
E' +|-id
E' + -|id
E' + -id|
E' + -E'|
E' + E'|
E' + E|
E|

(b)

|id + -id
id|+ -id
id +|-id
id + -|id
id + -id|
id + -E'|
id + E'|
id + E|
E' + E|
E|

(c)

|id + -id
|E' + -id
E'|+ -id
E' +|-id
E' + -|id
E' + -|E'
E' +|-E'
E' +|E'
E' +|E
E'|+ E
|E' + E
|E

(d)

|id + -id
id|+ -id
E'|+ -id
E' +|-id
E' + -|id
E' + -id|
E' + -E'|
E' + E'|
E' + E|
E|

9. Choose the grammar that correctly eliminates left recursion from the given grammar:

$E \longrightarrow E + T \mid T$
$T \longrightarrow id \mid (E)$

a) $E \longrightarrow E + id \mid E + (E) \mid id \mid (E)$
b) $E \longrightarrow TE'$
$E' \longrightarrow + TE' \mid \varepsilon$
$T \longrightarrow id \mid (E)$
c) $E \longrightarrow E' + T \mid T$
$E' \longrightarrow id \mid (E)$
$T \longrightarrow id \mid (E)$
d) $E \longrightarrow id + E \mid E + T \mid T$
$T \longrightarrow id \mid (E)$

10. For the given grammar, what is the correct series of reductions for the string: -(id + id) + id

$E \longrightarrow E' \mid E' + E$
$E' \longrightarrow -E' \mid id \mid (E)$

(a)
-(id + id) + id
-(id + E') + id
-(id + E) + id
-(E' + E) + id
-(E) + id
-E' + id
E' + id
E' + E'
E' + E
E

(b)
-(id + id) + id
-(E' + id) + id
-(E' + E') + id
-(E' + E') + E'
-(E' + E) + E'
-(E) + E'
-E' + E'
E' + E'
E' + E
E

(c)
-(id + id) + id
-(E' + id) + id
-(E' + E') + id
-(E' + E) + id
-(E) + id
-E' + id
E' + id
E' + E'
E' + E
E

(d)
-(id + id) + id
-(id + id) + E'
-(id + id) + E
-(E' + id) + E
-(E' + E') + E
-(E' + E) + E
-(E) + E
-E' + E
E' + E
E

Best wishes
Dr. Sherin El Gokhy